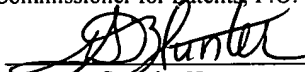


AF
JFW

CERTIFICATE OF MAIL

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail in an envelope addressed to Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on October 4, 2005.


Sandra Hunter

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In Re Application of:

Date: October 4, 2005

Matthew E. AUBERTINE

Confirmation No: 1980

Serial No: 09/903,937

Group Art Unit: 2192

Filed: July 12, 2001

Examiner: Fowlkes, Andre R.

For: METHOD AND SYSTEM FOR OPTIMIZING THE USE OF PROCESSORS
WHEN COMPILING A PROGRAM

APPEAL BRIEF

Joseph A. Sawyer, Jr.
Attorney for Appellant
Sawyer Law Group, LLP
2465 E. Bayshore Road, Suite 406
Palo Alto, CA 94303

10/07/2005 TBESHAH1 00000019 090447 09903937

01 FC:1402 500.00 DA

TABLE OF CONTENTS

I.	REAL PARTY IN INTEREST.....	1
II.	RELATED APPEALS AND INTERFERENCES.....	2
III.	STATUS OF CLAIMS.....	3
IV.	STATUS OF AMENDMENT.....	4
V.	SUMMARY OF CLAIMED SUBJECT MATTER.....	5
VI.	GROUND OF REJECTION TO BE REVIEWED ON APPEAL.....	6
VII.	ARGUMENTS.....	7
	A. Summary of the Applied Rejections.....	7
	B. The Cited Prior Art.....	8
	C. Independent claims 1, 7, and 13 are allowable over Gross.....	9
	D. Summary of Arguments.....	12
	APPENDIX A.....	i
	APPENDIX B (EVIDENCE - NONE).....	vii
	APPENDIX C (RELATED PROCEEDINGS – NONE)	viii

I. REAL PARTY IN INTEREST

Appellant respectfully submits that International Business Machines Corporation is the real party in interest.

II. RELATED APPEALS AND INTERFERENCES

Appellant states that no such proceeding exists.

III. STATUS OF CLAIMS

Claims 1-18 are pending and stand rejected. Accordingly, claims 1-18 are on appeal and all applied rejections concerning those claims are herein being appealed.

IV. STATUS OF AMENDMENT

Application Serial No. 09/903,937 (the instant application) as originally filed included claims 1-18. Claims 1-18 are pending. Claims 1-18 are on appeal and all applied prospective rejections concerning claims 1-18 are being appealed herein. All amendments made to the instant application have been entered.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention provides a method, system, and computer readable medium for optimizing the use of a plurality of processors when compiling a program in a computer system.

Independent claim 1 recites a method comprising: (a) locating a list of directories of the program and a list of processors of the computer system; (b) assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory; (c) compiling by the next available processor the at least one file within the next directory; and (d) repeating step (b) and (c) until there are no more directories to be compiled.

Independent claim 7 recites a system comprising: means for locating a list of directories in the program and a list of processors of the computer system; means for assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory; means for compiling by the next available at least one file within the next directory; and means for ensuring that there are no more directories to be compiled by utilizing the assigning means and the compiling means.

Independent claim 13 recites program instructions for: (a) locating a list of directories of the program and a list of processors of the computer system; (b) assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory; (c) compiling by the next available processor the at least one file within the next directory; and (d) repeating step (b) and (c) until there are no more directories to be compiled.

Support for independent claims 1, 7, and 13 is found in the combination of Figure 3; page 3, line 21, to page 4, line 9; and page 6, line 20, to page 7, line 10.

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Appellant respectfully seeks review of the following rejections:

1. Claims 1-18 are rejected under 35 U.S.C. 102(b) as being anticipated by Gross et al., (Gross), "Parallel Compilation for a Parallel Machine", ACM, 0-8791-306-X.

VII. ARGUMENTS

A. Summary of the Applied Rejections

The Final Office Action dated March June 6, 2005 rejected claims 1-18 under 35 U.S.C. 102(b) as being anticipated by Gross et al., (Gross), "Parallel Compilation for a Parallel Machine", ACM, 0-8791-306-X. In making the rejection, the Examiner stated:

Claims 1-18 are rejected under 35 U.S.C. 102(b) as being anticipated by Gross et al., (Gross), "Parallel Compilation for a Parallel Machine", ACM, 0-8791-306-X.

As per claim 1, Gross discloses a method for optimizing the use of a plurality of processors when compiling a program in a computer system, (p. 91 col. R:10-13, "Our need to speedup compilation ... (on) a parallel system, ... led us to investigate parallel compilation'), the method comprising the steps of:

(a) locating a list of directories of the program and a list of processors of the computer system, (p. 91 col. L:4-9, "We implemented a compiler that exploits parallelism by partitioning the input program for parallel translation ... and different functions of the application program are compiled in parallel on different workstations (i.e. different directories)", and p. 94 col. R:28-43, "The input to parallel make is a UNIX makefile (i.e. a list of directories of the program)", and p. 94 col. L:11-12, "The section masters (locate a list of processors and) attempt to distribute the function masters to different workstations (i.e. processors)",

(b) assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory (p. 94 col. L:56-58, "we adopt a simple first-come-first served strategy that distributes the tasks over the available processors"),

(c) compiling by the next available processor the at least one file within the next directory (p. 94 col. L: 56-58, "we adopt a simple first-come-first served strategy that distributes the tasks (i.e. files within the next directory) over the available processors (the next available processor)",

(d) repeating steps (b) and (c) until there are no more directories to be compiled (p. 94 col. L:56-58, "we adopt a simple first-come-first served strategy that distributes the tasks over the available processors"), this step essentially performs repeatedly, as processors finish their tasks and, again, become available)...

The Examiner stated the following in response to the previous arguments against these rejections:

Applicants arguments have been considered but they are not persuasive. In the remarks, the applicant has argued substantially that:

Gross does not disclose locating a list of directories as required by

amended claims 1, 7 and 13, at p. 9:11-10:17.

The Examiner's response:

The Gross reference discloses all of the limitations required by amended claims 1, 7 and 13 as addressed above in the art rejection.

Appellant respectfully requests that the Board reverse the Examiner's final rejection of the pending claims.

B. The Cited Prior Art

Gross discloses an application for a parallel computer with multiple, independent processors, which often includes different programs (functions) for the individual processors; compilation of such functions can proceed independently. The compiler exploits this parallelism by partitioning the input program for parallel translation. The host system for the parallel compiler is an Ethernet-based network of workstations, and different functions of the application program are compiled in parallel on different workstations. For typical programs in this environment, a speedup ranging from 3 to 6 using not more than 9 processors is observed. The paper includes detailed measurements for this parallel compiler; the system and implementation overhead is reported, as well as the speedup obtained when compared with sequential compilation. (Abstract.)

C. Independent claims 1, 7, and 13 are allowable over Gross

The present invention provides a method for optimizing the use of a plurality of processors when compiling a program in a computer system. The method and system comprises locating a list of directories of the program and a list of processors of the computer system. The method and system further includes assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory. The method and system also includes compiling by the next available processor the at least one file within the next directory. Finally, the method and system further includes ensuring there are no more directories to be compiled by repeating the assigning and compiling steps. Gross does not teach or suggest these features, as discussed below.

Gross does not teach or suggest the combination of “assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory,” “compiling by the next available processor the at least one file within the next directory,” and “repeating step (b) and (c) until there are no more directories to be compiled,” as recited in independent claims 1, 7, and 13.

The Examiner has referred to a simple first-come-first-served strategy that distributes tasks among available processors as teaching these steps. However, Gross states only **generally** that **tasks** are distributed among the available processors, but nowhere does Gross describe specifically **how** the tasks are distributed. Specifically, Gross states:

In our search we concentrated more on *parallelizing* compilation than on *scheduling* and *load balancing*, and we adopt a simple first-come-first-served strategy that **distributes the tasks over the available processors**. (Page 94, lines

56-58.)

As quoted above, Gross concentrates more on “parallelizing” compilation rather than on “scheduling and load balancing.” This high-level focus is further supported on page 94, line 39-41, where Gross states that the “parallel compiler implements rather **coarse grain** parallelism (the unit of work is the compilation of a function or section).” Gross does not provide the granularity of detail to teach or suggest assigning of a “next available directory to a next available processor” where the next available processor compiles “at least one file within the directory” or the other steps involving compiling the files within the directories, as recited in the present invention.

Furthermore, Gross is not describing parsed software that is to be compiled (e.g. function or section), but is instead describing compiling **tasks** that are distributed among available processors. Gross explicitly states that the “task” of a function master is to “implement phases 2 and 3 of the compiler, that is to optimize and generate code for one function” (page 93, lines 27-30). As stated above, the simple first-come-first-served strategy described by Gross “**distributes the tasks,**” not directories, over the available processors.

Furthermore, Gross does not teach or suggest, “locating a list of directories of the program and a list of processors of the computer system,” as recited in independent claims 1, 7, and 13. The Examiner has referred to page 91, lines 4-9, page 94, lines 11-12 and 28-43 as teaching this step. However, nowhere do these sections mention a “list of directories” of a program and a “list of processors” of a computer system as recited in the present invention. These sections merely describe general parallel compilation of modules. In rejecting the claims, the Examiner has contended that Gross discloses Applicant’s recited providing a list of directories and a list of processors by teaching “the number of processes on the function level, called function

masters is equal to the total number of functions in the program (i.e. directories containing parts of the code to be compiled) ... The section masters attempt to distribute the function masters to workstations (i.e. processors, from a list of processors).” Applicant fails to see how the use of function masters in Gross teaches or suggests the recited locating a list of directories.

Gross teaches on page 93 under the sub-heading “Function Level” that “the task of a function master is to implement phases 2 and 3 of the compiler, that is to optimize and generate code for one function.” There is nothing in this description of a function master to optimize and generate code for one function that teaches or suggests anything regarding a directory. As described by the Applicant in the first paragraph of page 2 of the specification, “to more efficiently compile the complex program from version to version in the development process, the program is broken up into a directory structure where discrete files of the code are placed in specific directories or subdirectories. Within each directory are files related to that particular file of the code.” Thus, as is well understood in the art, a directory provides a level in a hierarchy of data related to the code being compiled, i.e., it contains files and/or subdirectories of discrete files of code. The nature of the recited directory containing at least one file is demonstrated in the recited assigning of a directory to a next processor in an ordered manner to allow the next available processor to compile at least one file within the directory.

In contrast, the so-called “directories” of the function masters in Gross are in and of themselves singular entities of code for one function. While the Examiner has asserted that function masters in Gross correspond with the recited directories, given the description of Gross that function masters actually optimize and generate code for one function. Applicant respectfully submits that these function masters provide a singular level of data that is not associated with or capable of indicating a hierarchy, as done by Applicant’s recited directory

within which is at least one file.

In view of the foregoing, Applicant respectfully submits that the function masters in Gross offer no teaching or suggestion of a list of directories, as contended by the Examiner. Without teaching or suggesting the provision of a list of directories in Gross, there is nothing to teach or suggest the invention, as recited in independent claims 1, 7, and 13.

Therefore, Gross does not teach or suggest the combination of steps as recited in independent claims 1, 7, and 13, and these claims are allowable over Gross.

Dependent claims 2-6, 8-12, and 14-18 depend from independent claims 1, 7, and 13, respectively. Accordingly, the above-articulated arguments related to independent claims 1, 7, and 13 apply with equal force to claims 2-6, 8-12, and 14-18, which are thus allowable over the cited reference for at least the same reasons as claims 1, 7, and 13.

In view of the foregoing, Applicant respectfully submits that the recited invention is not taught, shown, or suggested by the cited art.

Accordingly, Appellant respectfully requests withdrawal of the rejection under 35 U.S.C. 102(b) and respectfully requests that the Board reverse the final rejection of claims.

D. Summary of Arguments

For all the foregoing reasons, it is respectfully submitted that claims 1-18 (all the claims presently in the application) are patentable for defining subject matter, which would not have been unpatentable under 35 U.S.C. 102(b) at the time the subject matter was invented. Thus, Appellant respectfully requests that the Board reverse the rejection of all the appealed claims and find each of these claims allowable.

Note: For convenience of detachment without disturbing the integrity of the remainder of pages of this Appeal Brief, Appellant's APPENDICES A-C are attached on separate sheets following the signatory portion of this Appeal Brief.

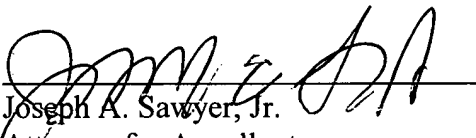
Please charge any fee that may be necessary for the continued pendency of this application to Deposit Account No. 09-0460 (IBM Corporation).

Respectfully submitted,

SAWYER LAW GROUP LLP

October 4, 2005

Date


Joseph A. Sawyer, Jr.
Attorney for Appellant
Reg. No. 30,801
(650) 493-4540

APPENDIX A

CLAIMS

1. (Previously presented) A method for optimizing the use of a plurality of processors when compiling a program in a computer system, the method comprising the steps of:
 - (a) locating a list of directories of the program and a list of processors of the computer system;
 - (b) assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory;
 - (c) compiling by the next available processor the at least one file within the next directory;and
 - (d) repeating step (b) and (c) until there are no more directories to be compiled.
2. (Original) The method of claim 1 wherein the assigning step (b) further includes the step of (b1) obtaining a directory in which all dependencies have been satisfied.
3. (Original) The method of claim 1 wherein the assigning step (b) further includes the step of (b1), updating the list of processors and the list of directories based upon the assignment of the directory.
4. (Original) The method of claim 1 wherein the assigning step (b) further includes the step of (b1) providing a directory update mechanism for assigning the directories in the ordered manner.

5. (Original) The method of claim 4 wherein the providing an update mechanism step (b1) further comprises the steps of:

(b11) providing an array of dependency changes; and

(b12) merging the dependency changes array with a master array of changes.

6. (Original) The method of claim 5 wherein the merging step (b12) comprises the steps of:

(b121) obtaining a dependency change from the dependency changes array;

(b122) determining whether the dependency change is in a directory in the master array;

(b123) updating the directory in the master array of the dependency change in a directory of the master array;

(b124) adding dependency change to the master array in a new directory if the dependency change is not in a directory of the master array;

(b125) determining if there is another dependency change in the dependency changes array after either step (b123) or step (b124); and

(b126) repeating steps (b121) – (b125) until all dependency changes have been obtained from the dependency change array.

7. (Previously presented) A system for optimizing the use of a plurality of processors when compiling a program in a computer system, the system comprising:

means for locating a list of directories in the program and a list of processors of the computer system;

means for assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory;

means for compiling by the next available at least one file within the next directory; and

means for ensuring that there are no more directories to be compiled by utilizing the assigning means and the compiling means.

8. (Original) The system of claim 7 wherein the assigning means further includes means for obtaining a directory in which all dependencies and prerequisites have been satisfied.

9. (Original) The system of claim 7 wherein the assigning means further includes the means for updating the list of processors and the list of directories based upon the assignment of the directory.

10. (Previously presented) The system of claim 7 wherein the assigning means further includes the means for providing a directory update mechanism for assigning the directories in the ordered manner.

11. (Original) The system of claim 10 wherein the providing an update mechanism means further comprises:

means for providing an array of dependency changes; and

means for merging the dependency changes array with a master array of changes.

12. (Previously presented) The system of claim 11 wherein the merging means comprises:

means for obtaining a dependency change from the dependency changes array;

means for determining whether the dependency change is in a directory in the master array;

means for updating the directory in the master array of the dependency change in a directory of the master array;

means for adding dependency change to the master array in a new directory if the dependency change is not in a directory of the master array;

means for determining if there is another dependency change in the dependency changes array;

and

means for ensuring that all dependency changes have been obtained from the dependency change array.

13. (Previously presented) A computer readable medium containing program instructions for optimizing the use of a plurality of processors when compiling a program in a computer system, the program instructions for:

(a) locating a list of directories of the program and a list of processors of the computer system;

(b) assigning a next directory to a next available processor in an ordered manner to allow the next available processor to compile at least one file within the directory;

(c) compiling by the next available processor the at least one file within the next directory; and

(d) repeating step (b) and (c) until there are no more directories to be compiled.

14. (Original) The computer readable medium of claim 13 wherein the assigning step (b)

further includes the step of (b1) obtaining a directory in which all dependencies and prerequisites have been satisfied.

15. (Previously presented) The computer readable medium of claim 13 wherein the assigning step (b) further includes the step of (b1), updating the list of processors and the list of directories based upon the assignment of the directory.

16. (Original) The computer readable medium of claim 13 wherein the assigning step (b) further includes the step of (b1) providing a directory update mechanism for assigning the directories in the ordered manner.

17. (Original) The computer readable medium of claim 16 wherein the providing an update mechanism step (b1) further comprises the steps of:

(b11) providing an array of dependency changes; and

(b12) merging the dependency changes array with a master array of changes.

18. (Original) The computer readable medium of claim 17 wherein the merging step (b12) comprises the steps of:

(b121) obtaining a dependency change from the dependency changes array;

(b122) determining whether the dependency change is in a directory in the master array;

(b123) updating the directory in the master array of the dependency change in a directory of the master array;

(b124) adding dependency change to the master array in a new directory if the dependency

change is not in a directory of the master array;

(b125) determining if there is another dependency change in the dependency changes array after either step (b123) or step (b124); and

(b126) repeating steps (b121) – (b125) until all dependency changes have been obtained from the dependency change array.

APPENDIX B

EVIDENCE

(NONE)

APPENDIX C
RELATED PROCEEDINGS
(NONE)

TRANSMITTAL FORM

Attorney Docket No.
AUS920000329US1
1753P

In re the application of: **AUBERTINE P E** Confirmation No: **1980**

Serial No: **09/903,937**

Group Art Unit: **2192**

Filed: **July 12, 2001**

Examiner: **Fowlkes, Andre R.**

For: **Method and System for Optimizing The Use of Processors When Compiling A Program**

ENCLOSURES (check all that apply)

<input type="checkbox"/>	Amendment/Reply	<input type="checkbox"/>	Assignment and Recordation Cover Sheet	<input type="checkbox"/>	After Allowance Communication to Group
<input type="checkbox"/>	After Final	<input type="checkbox"/>	Part B-Issue Fee Transmittal	<input type="checkbox"/>	Notice of Appeal
<input type="checkbox"/>	Information disclosure statement	<input type="checkbox"/>	Letter to Draftsman	<input checked="" type="checkbox"/>	Appeal Brief
<input type="checkbox"/>	Form 1449	<input type="checkbox"/>	Drawings	<input type="checkbox"/>	Status Letter
<input type="checkbox"/>	(X) Copies of References	<input type="checkbox"/>	Petition	<input checked="" type="checkbox"/>	Postcard
<input type="checkbox"/>	Extension of Time Request *	<input type="checkbox"/>	Fee Address Indication Form	<input type="checkbox"/>	Other Enclosure(s) (please identify below):
<input type="checkbox"/>	Express Abandonment	<input type="checkbox"/>	Terminal Disclaimer		
<input type="checkbox"/>	Certified Copy of Priority Doc	<input type="checkbox"/>	Power of Attorney and Revocation of Prior Powers		
<input type="checkbox"/>	Response to Incomplete Appln	<input type="checkbox"/>	Change of Correspondence Address		
<input type="checkbox"/>	Response to Missing Parts	*Extension of Term: Pursuant to 37 CFR 1.136, Applicant petitions the Commissioner to extend the time for response for xxxxxx month(s), from to .			
<input type="checkbox"/>	Executed Declaration by Inventor(s)				

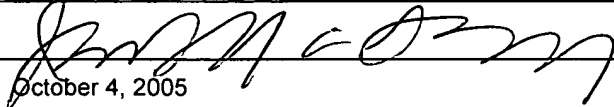
CLAIMS

FOR	Claims Remaining After Amendment	Highest # of Claims Previously Paid For	Extra Claims	RATE	FEE
Total Claims	0	0	0	\$ 50.00	\$ 0.00
Independent Claims	0	0	0	\$200.00	\$ 0.00
				Total Fees	\$ 0.00

METHOD OF PAYMENT

<input type="checkbox"/>	Check no. _____ in the amount of \$ _____ is enclosed for payment of fees.
<input checked="" type="checkbox"/>	Charge \$500.00 to Deposit Account No. <u>09-0447</u> (IBM Corporation) for payment of fees.
<input checked="" type="checkbox"/>	Charge any additional fees or credit any overpayment to Deposit Account No. <u>09-0447</u> (IBM Corporation)

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Attorney Name	Joseph A. Sawyer, Jr., Reg. No. 30,801
Signature	
Date	October 4, 2005

CERTIFICATE OF MAILING

I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Mail Stop Appeal Brief-Patents, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on October 4, 2005	
Type or printed name	Sandra D. Hunter
Signature	